IN THE CLAIMS:

Please cancel claims 1-29 and add new claims 30-55.

1-29. (Cancelled)

30. (New) A computer-implemented method for debugging in a distributed debugging environment, comprising:

displaying, on a display interface, a value of one or more data variables of a program being debugged;

receiving a command to execute the program being debugged;

in response to receiving the command, executing the program from a first execution point to a second execution point;

determining, based on the first and second execution points, which data variables may have been modified by the execution of the program; and

refreshing the value displayed on the display interface only for data variables that may have been modified by the execution of the program from the first execution point to the second execution point.

31. (New) The method of claim 30, wherein the command comprises a step command configured to cause the execution of a statement of the program being debugged present at the first execution point.

32. (New) The method of claim 31, wherein executing the program to a second execution point, comprises:

executing the statement; and

determining a set of variables that may have been modified by the execution of the statement.

33. (New) The method of claim 30, wherein the command comprises a run command, and wherein the second execution point comprises a breakpoint encountered by the execution of the program from the first execution point.

34. (New) The method of claim 33, wherein executing the program to the second execution point, comprises:

if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, determining a set of variables that may be modified by the execution of each statement of the node from the first execution point to the breakpoint; and

executing each statement of the node from the first execution point to the breakpoint.

35. (New) The method of claim 33, further comprising,

if the node does not contain a breakpoint between the first execution point and a last statement of the node, then beginning with a root node, marking each node of the control flow graph which may be exited and reentered during execution of the program from the first execution point;

for each marked node, marking all data variables of the node as variables that may be modified by the execution of the program from the first execution point to the second execution point; and

beginning with the node containing the first execution point, generating a list of unmarked nodes which may be reached during execution of the program from the first execution point.

36. (New) The method of claim 35, wherein generating the list comprises:

traversing the control flow graph from the node containing the first execution point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

37. (New) The method of claim 35, further comprising:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all data variables associated with all statements of the preceding

unmarked node and all propagated kill variables propagated to the preceding unmarked node.

38. (New) The method of claim 30, wherein the debugger interface is displayed on a first computer system, and wherein the program being debugged is executing on a second computer system, and wherein the first and second computer systems are configured to communicate over a network communication channel.

39. (New) A computer-readable medium containing a program which, when executed, performs an operation method for maximizing debugger performance in a distributed debugging environment, comprising:

displaying, on a display interface, a value of one or more data variables of a program being debugged;

receiving a command to execute the program being debugged;

in response to receiving the command, executing the program from a first execution point to a second execution point;

determining, based on the first and second execution points, which data variables may have been modified by the execution of the program; and

refreshing the value displayed on the display interface only for data variables that may have been modified by the execution of the program from the first execution point to the second execution point.

40. (New) The computer-readable medium of claim 39, wherein the command comprises a step command configured to cause the execution of a code statement of the program being debugged present at the first execution point.

41. (New) The computer-readable medium of claim 40, wherein executing the program to a second execution point, comprises:

executing the statement; and

determining a set of variables that may have been modified by the execution of the statement.

42. (New) The computer-readable medium of claim 39, wherein the command comprises a run command, and wherein the second execution point comprises a breakpoint encountered by the execution of the program from the first execution point.

43. (New) The computer-readable medium of claim 42, wherein executing the program to a second execution point, comprises:

if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, determining a set of variables that may be modified by the execution of each statement of the node from the first execution point to the breakpoint; and

executing each statement of the node from the first execution point to the breakpoint.

44. (New) The computer-readable medium of claim 42, wherein the operation further comprises:

if the node does not contain a breakpoint between the first execution point and the last statement of the node, then beginning with a root node, marking each node of the control flow graph which may be exited and reentered during execution of the program from the first execution point;

for each marked node, marking all data variables of the node as variables that may be modified by the execution of the program from the first execution point to the second execution point; and

beginning with the node containing the first execution point, generating a list of unmarked nodes which may be reached during execution of the program from the first execution point.

45. (New) The computer-readable medium of claim 43, wherein generating the list comprises:

traversing the control flow graph from the node containing the first execution point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

46. (New) The computer-readable medium of claim 43, wherein the operation further comprises:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all data variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.

47. (New) The computer-readable medium of claim 39, wherein the debugger interface is displayed on a first computer system, and wherein the program being debugged is executing on a second computer system, and wherein the first and second computer systems are configured to communicate over a network communication channel.

48. (New) A computing device, comprising:

a processor;

a memory; and

a debugger program which when executed on the processor, performs a method for maximizing debugger performance in a distributed debugging environment, comprising:

displaying, on a display interface, a value of one or more data variables of a program being debugged;

receiving a command to execute the program being debugged;

in response to receiving the command, executing the program from a first execution point to a second execution point;

determining, based on the first and second execution points, which data variables may have been modified by the execution of the program; and

refreshing the value displayed on the display interface only for data variables that may have been modified by the execution of the program from the first execution point to the second execution point.

49. (New) The computing device of claim 48, wherein the command comprises a step command configured to cause the execution of a code statement of the program being debugged present at the first execution point.

50. (New) The computing device of claim 49, wherein executing the program to a second execution point, comprises:

executing the statement; and

determining a set of variables that may have been modified by the execution of the statement.

51. (New) The computing device of claim 48, wherein the command comprises a run command, and wherein the second execution point comprises a breakpoint encountered by the execution of the program from the first execution point.

52. (New) The computing device of claim 51, wherein executing the program to a second execution point, comprises:

if a node of a control flow graph that contains the first execution point of the program contains a breakpoint positioned after the first execution point, determining a set of variables that may be modified by the execution of each statement of the node from the first execution point to the breakpoint; and

executing each statement of the node from the first execution point to the breakpoint.

53. (New) The computing device of claim 51, wherein the operation further comprises:

if the node does not contain a breakpoint between the first execution point and the last statement of the node, then beginning with a root node, marking each node of the control flow graph which may be exited and reentered during execution of the program from the first execution point;

for each marked node, marking all data variables of the node as variables that may be modified by the execution of the program from the first execution point to the second execution point; and

beginning with the node containing the first execution point, generating a list of unmarked nodes which may be reached during execution of the program from the first execution point.

54. (New) The computer-readable medium of claim 53, wherein generating the list comprises:

traversing the control flow graph from the node containing the first execution point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

55. (New) The computing devices of claim 53, wherein the operations fo the program further comprise:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all data variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.